Reinforcement learning project

Jere Knuutinen - 994349
Juuso Korhonen - 652377
ELEC-E8125 - Reinforcement Learning

December 14, 2022

1 PART I

Task 1

Question 1: Which algorithm performed better in terms of averaged episode reward? What might be the reason in your opinion? Compare algorithms using sample-efficiency and clock-time.

Hopper medium enviroment

Algorithms that we used were DDPG and PG. When comparing the these two in terms of average episode reward DDPG performed better. Figure 1 shows the average episode reward in the case of PG algorithm.

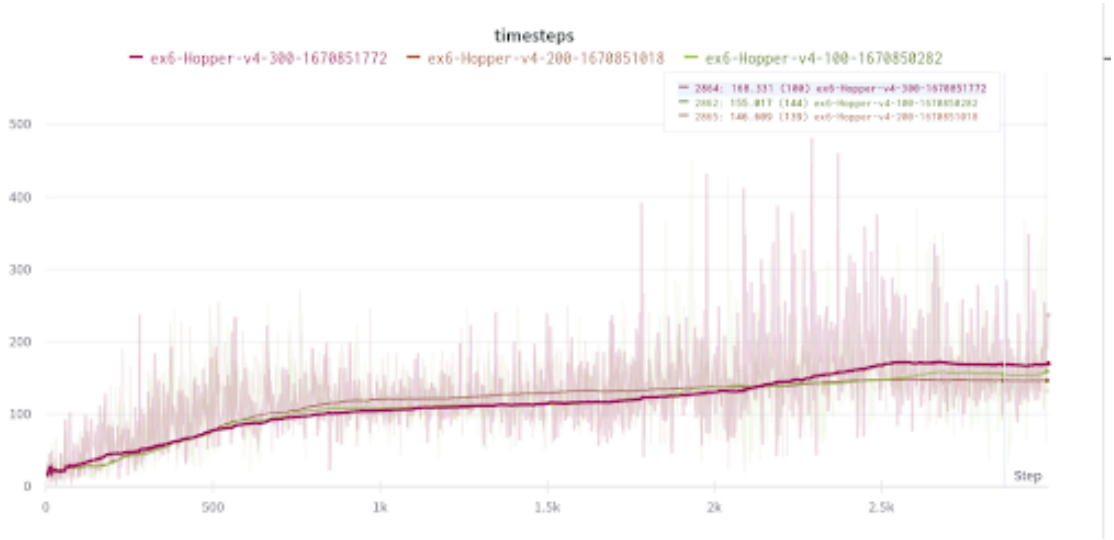

Figure 1: Smoothed episode reward with the PG algorithm, ran with 3 different seeds. Enviroment hopper-medium

Seed 100	Seed 200	Seed 300	Mean/std
2778.6	2989.1	2675.0	2714.2/130.6

Table 1: Mean test episode rewards for 100 episodes with trained DDPG algorithm, trained with 3 different seeds. Enviroment hopper-medium

As one can see from the figure 1, the results with PG algorithm were not promising despite the fact that multiple hyperparameter settings were tried. Possible reasons are:

- High variance in the policy gradient estimate due to stochastic policy, slow convergence.
- On-policy method, not possible use replay buffer.

These are all features of PG that make the training hard.

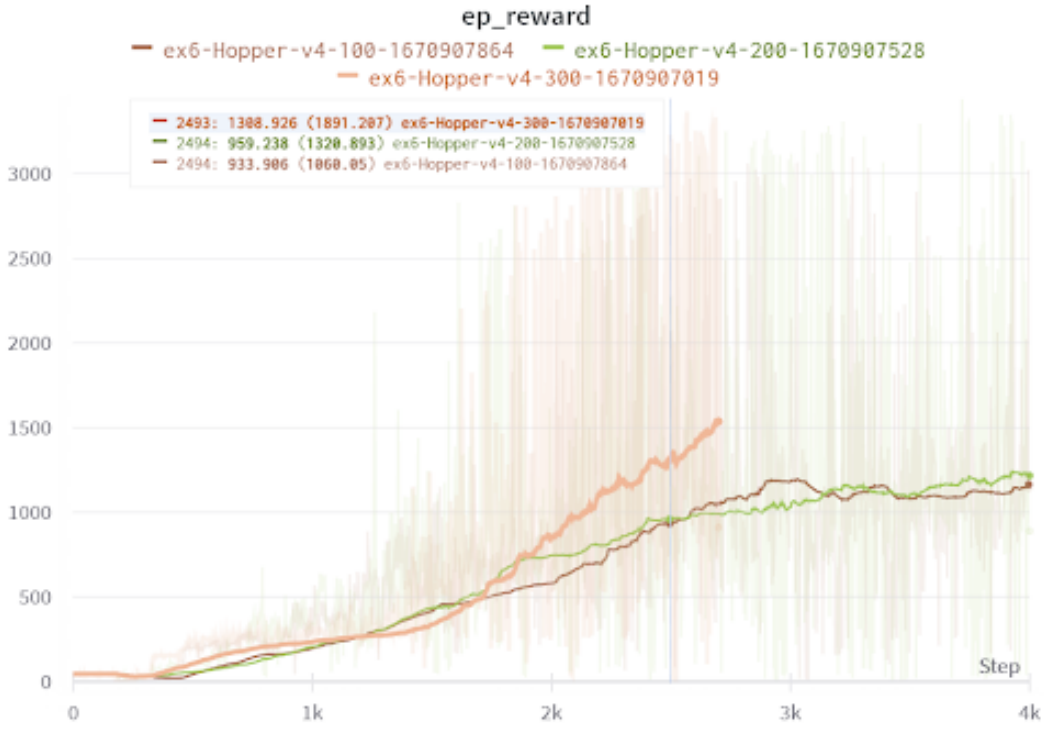

Figure 2: Smoothed episode reward with the DDPG algorithm, ran with 3 different seeds. There was an early stopping condition, that was fulfilled when the mean reward for 100 test episodes was greater than 2000. Enviroment hopper-medium

As one can see from results in figure 2 and table 1, the DDPG algorithm performed better than PG. Possible reasons are:

- use of critic decreases the variance of the policy gradient and can make the learning faster.

- off-policy method so allows the use of replay buffer.

Figure 3 shows both PG and DDPG results in one plot.

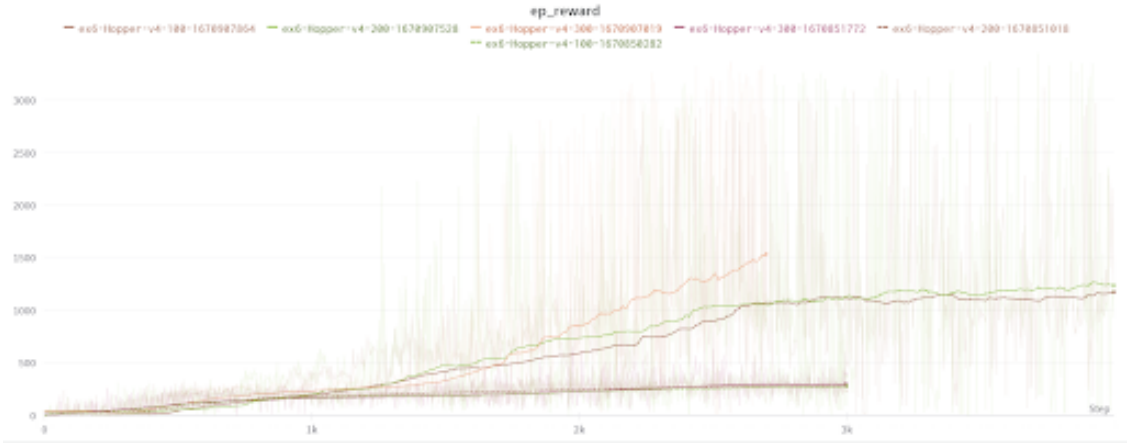

Figure 3: Smoothed episode reward with the DDPG algorithm on Hopper medium environment, ran with 3 different seeds. There was an early stopping condition, that was fulfilled when the mean reward for 100 test episodes was greater than 2000. Also PG agent only ran 3000 episodes.

Lunar lander medium enviroment

Figure 5 depicts PG performance on Lunar lander medium environment. One can see that PG does not perform well in this environment. The reason why PG agent does not perform well is explained in the previous section.

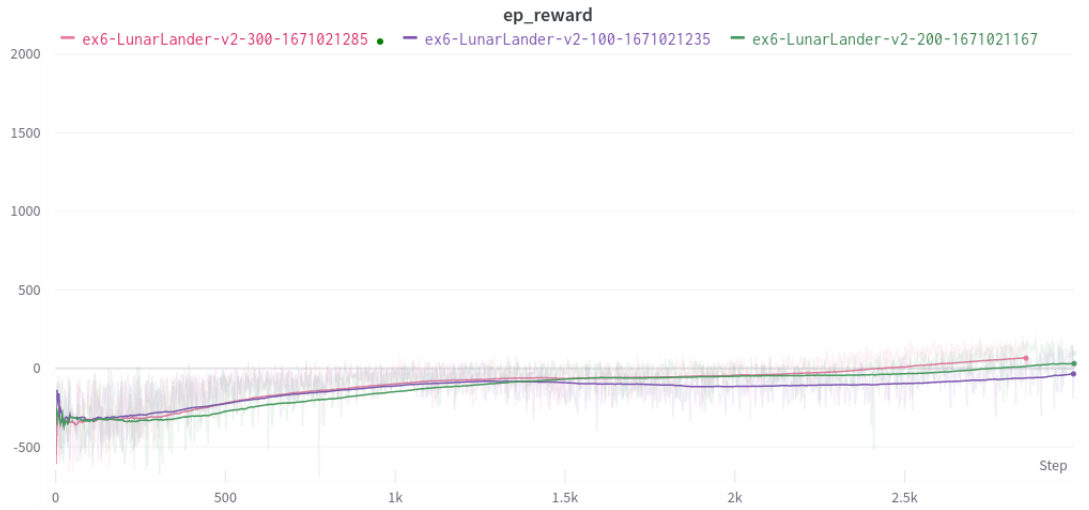

Figure 4: Smoothed episode reward with the PG algorithm, ran with 3 different seeds. Enviroment Lunar-lander-medium

Seed 100	Seed 200	Seed 300	Mean/std
125.1	130.5	202.2	152.3/35

Table 2: Mean test episode rewards for 100 episodes with trained DDPG algorithm, trained with 3 different seeds. Enviroment hopper-medium

Figure 5 depicts DDPG performance on Lunar lander medium enviroment. Again we observe better performance compared to PG. Mean test performance is depicted in table 2.

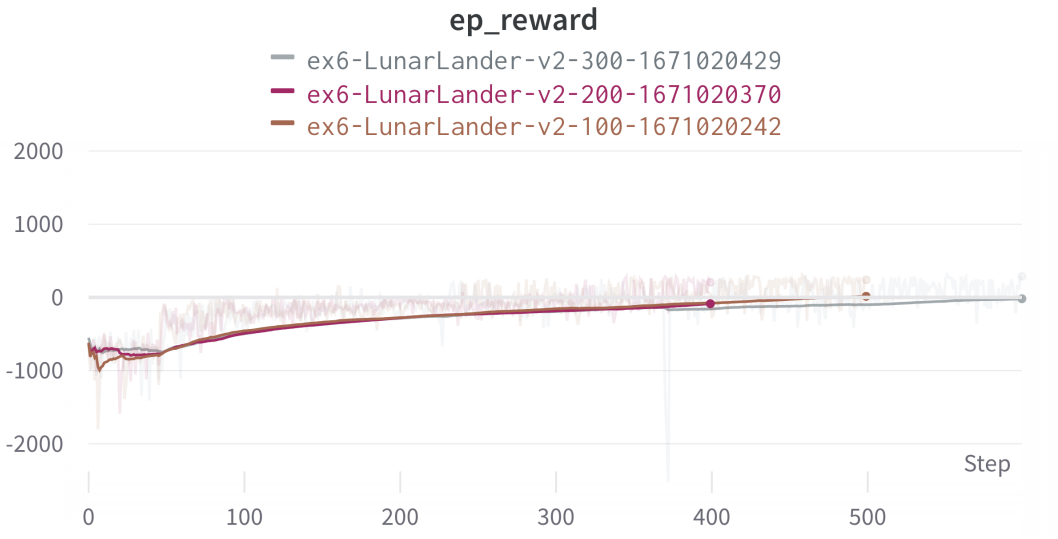

Figure 5: Smoothed episode reward with the DDPG algorithm, ran with 3 different seeds. There was an early stopping condition, that was fulfilled when the mean reward for 100 test episodes was greater than 100. Enviroment Lunar lander medium

As 6 shows sample efficiency is much better in the case of DDPG, however clock time was better in the case of PG since they ran almost same time. Again reasons for these behaviours are that in the case of DDPG we basically train two neural nets where as in the case PG we only train one neural net and thats why clock time is better in PG. Reason for improvement in sample efficiency with DDPG due to the use of critic is already stated in hopper environment case.

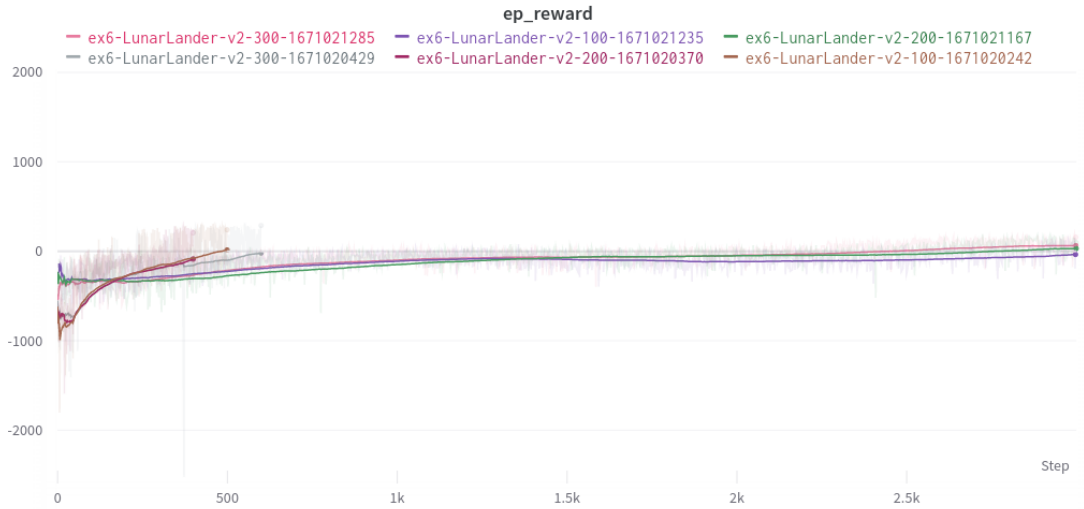

Figure 6: Smoothed episode reward with the DDPG algorithm on Hopper medium environment, ran with 3 different seeds. There was an early stopping condition, that was fulfilled when the mean reward for 100 test episodes was greater than 100.

Question 2 Which hyperparameters did you change to make the algorithm work? Why? Justify your answer with graphical and algorithmic arguments. (5')

Buffer size and action noise (exploration) were increased with the DDPG.

Buffer size:

Increasing the replay buffer size makes you less likely to sample correlated samples, which makes the training of the neural networks more stable. Larger size also makes the network less prone to catastrophic forgetting, as small buffer might force network to make inference of only small timescale.

Action noise:

Adding action noise helps to explore the action space.. With too little action noise, the actor might converge too quickly to follow the deterministic critic.

2 PART II

Question 1 Select a single research paper from the list of research papers above. Read the paper and explain the main ideas of the paper and the intuition behind it.

In the paper "Continuous Deep Q-Learning with Model-based Acceleration" the authors explore ways to reduce the sample complexity of deep reinforcement learning for continuous control tasks. For this purpose, they represent a variant of Q-learning called normalized advantage functions (NAF) suitable for continuous domains. They also explore the use of

learned models, but we can skip this part of the paper, as they do not discover major benefits with this approach and this is not part of this project's and course scope.

Model-free reinforcement learning in domains with continuous actions is typically handled with policy search methods. Integrating value function estimation to these methods has been previously implemented with the use of actor-critic methods, but this already requires training two separate function approximators which can require huge amounts of samples. The authors of the paper propose to combine the estimation of advantage and value function to one neural network. This simple architecture is presented to make the learning more efficient.

The developed method in the paper is normalized advantage function (NAF). In NAF, there are three output channels from single neural network, one estimates value function ($V(x|\theta^V)$), one estimates lower triangular matrix ($L(x|\theta^P)$) and one estimates action that basically maximizes Q function ($\mu(x|\theta^\mu)$). With these outputs, Q function can be calculated as follows:

$$Q(x|\theta^Q) = A(x, u|\theta^A) + V(x|\theta^V) \tag{1}$$

In turn, advantage as follows

$$A(x, u|\theta^A) = -0.5(u - \mu(x|\theta^\mu)^T P(x|\theta^P)(u - \mu(x|\theta^\mu) \tag{2}$$

where $P(x|\theta^P) = L(x|\theta^P)L(x|\theta^P)^T$

Question 2 Did the implemented ideas improve the baseline performance in terms of episode reward, sample-efficiency, or clock-time? Please justify your answer using graphical arguments and numerical information from your experiments, e.g. episode reward plot. (5')

We couldn't get the NAF algorithm improve or even reach the DDPG baseline. The training reward seems to get stuck at 200. Despite extensive debugging even with the course assistants, we weren't able to find the reason for this behaviour.

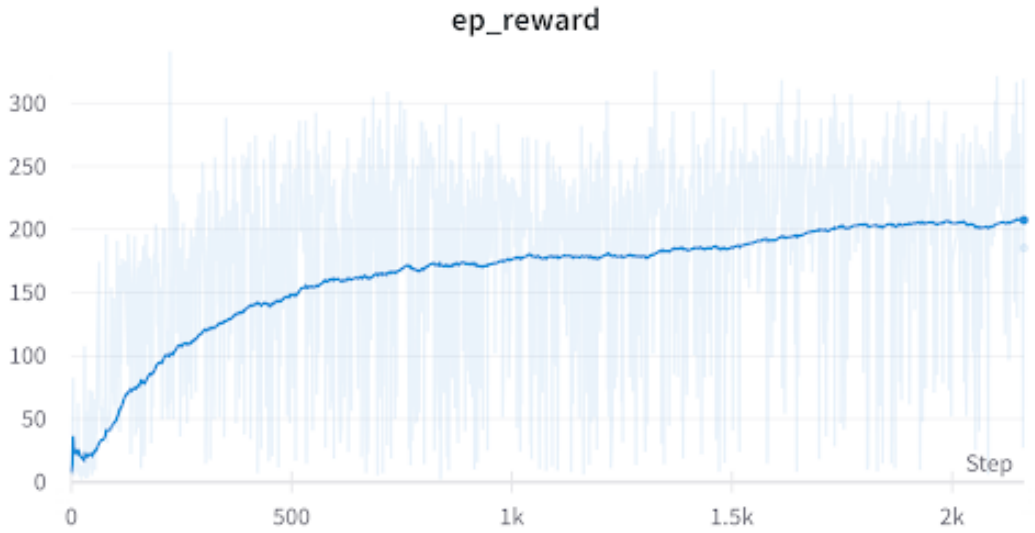

Figure 7: Smoothed episode reward with the DDPG algorithm, ran with 3 different seeds. There was an early stopping condition, that was fulfilled when the mean reward for 100 test episodes was greater than 2000.

Question 3 Can upgraded algorithms cope with increased environment complexity? Why / why not do you think the proposed approaches should improve the baseline algorithms? (5')

NAF, when implemented correctly, can most likely cope with increased environment complexity. One can for example scale the network complexity (layer sizes, skip connections etc.) along with environment. We also think that correctly implemented NAF would improve over DDPG, since it has more efficient structure due to combining the actor and the critic part onto one network.

4.7 Extra task If you're doing good and have the energy and passion for the more serious environment using algorithms from Part II, then try your algorithm to solve the environment from $hopper_{h}ard.yaml.Forthis$

References